CLAIMS

1. (Currently Amended) In a wireless telephone ~~communications device~~, a method for executing dynamic instruction sets, the method comprising:

executing system software;

launching a run-time engine;

processing dynamic instruction sets;

operating on system data and system software; and,

in response to operating on the system data and system software, controlling the execution of the system software.

2. (Original) The method of claim 1 further comprising:

following the processing of the dynamic instruction sets, deleting dynamic instruction sets.

3. (Original) The method of claim 1 wherein processing dynamic instruction sets includes processing instructions in response to mathematical and logical operations.

4. (Original) The method of claim 3 further comprising:

receiving the dynamic instruction sets.

5. (Original) The method of claim 4 wherein receiving the dynamic instruction sets includes receiving the dynamic instruction sets through an interface selected from the group including airlink, radio frequency (RF) hardline, installable memory module, infrared, and logic port interfaces.

6. (Original) The method of claim 5 further comprising:

forming the system software into symbol libraries, each symbol library comprising symbols having related functionality;

arranging the symbol libraries into code sections; and,

wherein launching a run-time engine includes invoking a run-time library from a first code section.

7. (Original) The method of claim 6 wherein receiving the dynamic instruction set includes receiving a patch manager run time instruction (PMRTI) in a file system section nonvolatile memory.

8. (Original) The method of claim 7 wherein receiving the patch manager run time instructions includes receiving conditional operation code and data items;

wherein processing dynamic instruction sets includes:

using the run-time engine to read the patch manager run time instruction operation code; and,

performing a sequence of operations in response to the operation code.

9. (Original) The method of claim 8 wherein processing dynamic instruction sets includes:

using the run-time engine to capture the length of the patch manager run time instruction;

extracting the data items from the patch manager run time instruction, in response to the operation code; and,

using the extracted data in performing the sequence of operations responsive to the operation code.

10. (Original) The method of claim 9 wherein arranging the symbol libraries into code sections includes starting symbol libraries at the start of code sections and arranging symbols to be offset from their respective code section start addresses;

the method further comprising:

storing the start of code sections at corresponding start addresses;

maintaining a code section address table cross-referencing code section identifiers with corresponding start addresses; and,

maintaining a symbol offset address table cross-referencing symbol identifiers with corresponding offset addresses, and corresponding code section identifiers.

11. (Original) The method of claim 10 wherein receiving the patch manager run time instruction includes receiving symbol identifiers;

the method further comprising:

locating symbols corresponding to the received symbol identifiers by using the code section address table and symbol offset address table;

wherein performing a sequence of operations in response to the operation code includes:

when the located symbols are data items, extracting the data; and,

when the located symbols are instructions, executing the symbols.

12. (Original) The method of claim 8 wherein processing dynamic instruction sets includes:

accessing system data stored in a second code section in the file system section;

analyzing the system data;

creating updated system data;

wherein operating on system data and system software includes replacing the system data in the second section with the updated system data; and,

wherein controlling the execution of the system software includes using the updated system data in the execution of the system software.

13. (Original) The method of claim 8 further comprising:

storing a plurality of code sections in a code storage section nonvolatile memory;

wherein processing dynamic instruction sets includes:

accessing system data stored in a third code section in the code storage section;

analyzing the system data;

creating updated system data;

wherein operating on the system data and system software includes replacing the system data in the third code section with the updated system data; and,

wherein controlling the execution of the system software includes using the updated system data in the execution of the system software.

14. (Original) The method of claim 8 further comprising:

storing a plurality of code sections in a code storage section nonvolatile memory;

loading read-write data into volatile memory;

wherein processing dynamic instruction sets includes:

accessing the read-write data in volatile memory;

analyzing the read-write data;

creating updated read-write data;

wherein operating on the system data and system software includes replacing the read-write data in volatile memory with the updated read-write data; and,

wherein controlling the execution of the system software includes using the updated read-write data in the execution of the system software.

15. (Original) The method of claim 8 wherein processing dynamic instruction sets includes:

in response to the operation code, monitoring the execution of the system software;

collecting performance data;

storing the performance data; and,

wherein operating on the system data and system software includes using the performance data in the evaluation of system software.

16. (Original) The method of claim 15 further comprising:

transmitting the stored data via an airlink interface.

17. (Original) The method of claim 8 further comprising:

storing a plurality of code sections in a code storage section nonvolatile memory;

wherein receiving patch manager run time instructions includes receiving a new code section;

wherein operating on the system data and system software includes adding the new code section to the code storage section; and,

wherein controlling the execution of the system software includes using the new code section in the execution of the system software.

18. (Original) The method of claim 17 wherein receiving a new code section includes receiving an updated code section; and,

wherein operating on the system data and system software includes replacing a fourth code section in the code storage section with the updated code section.

19. (Currently Amended) In a wireless telephone ~~communications device~~, a method for executing dynamic instruction sets, the method comprising:

forming the system software into symbol libraries, each symbol library comprising symbols having related functionality;

arranging the symbol libraries into code sections in a code storage section nonvolatile memory;

executing system software;

receiving a patch manager run time instruction (PMRTI), including conditional operation code and data items, in a file system section nonvolatile memory;

calling a run-time library from a first code section;

processing the patch manager run time instruction operation code;

operating on system data and system software; and,

in response to operating on the system data and system software, controlling the execution of the system software.

20. (Currently Amended) In a wireless telephone ~~communications device~~, a dynamic instruction set execution system, the system comprising:

executable system software and system data differentiated into code sections;

dynamic instruction sets for operating on the system data and the system software, and controlling the execution of the system software; and,

a run-time engine for processing the dynamic instruction sets.

21. (Original) The system of claim 20 wherein the run-time engine processes dynamic instruction sets to perform mathematical and logical operations.

22. (Original) The system of claim 21 further comprising:

a file system section nonvolatile memory for receiving the dynamic instruction sets.

23. (Original) The system of claim 22 further comprising:

an interface through which the dynamic instruction sets are received into the file system section, wherein the interface is selected from the group including airlink,

radio frequency (RF) hardline, installable memory module, infrared, and logic port interfaces.

24. (Original) The system of claim 23 wherein the executable system software and system data include symbol libraries, each symbol library comprising symbols having related functionality, arranged into code sections; and,

wherein the run-time engine is a run-time library arranged in a first code section.

25. (Original) The system of claim 24 wherein the dynamic instruction sets include conditional operation code and data items, and wherein the dynamic instruction sets are organized in a patch manager run time instruction (PMRTI).

26. (Original) The system of claim 25 further comprising:

a code storage section nonvolatile memory for storing code sections.

27. (Original) The system of claim 26 wherein the run-time engine reads the dynamic instruction set operation code and performs a sequence of operations in response to the operation code.

28. (Original) The system of claim 27 wherein the run-time engine captures the length of a dynamic instruction set to determine if data items are included, extracts the data items from the dynamic instruction set, and uses the extracted data in performing the sequence of operations responsive to the operation code.

29. (Original) The system of claim 28 wherein the symbol libraries are arranged to start at the start of code sections and symbols are arranged to be offset from their respective code section start addresses;

wherein a code storage section includes start addresses corresponding to code section start addresses;

the system further comprising:

a code section address table cross-referencing code section identifiers with corresponding start addresses in the code storage section; and,

a symbol offset address table cross-referencing symbol identifiers with corresponding offset addresses, and corresponding code section identifiers.

30. (Original) The system of claim 27 wherein the dynamic instruction set includes symbol identifiers; and,

wherein the run-time engine locates symbols corresponding to the received symbol identifiers using the code section address table and symbol offset address table, extracts data when the located symbols are data items, and executes the symbols when the located symbols are instructions.

31. (Original) The system of claim 27 wherein the system data is stored in a second code section in the file system section;

wherein the run-time engine accesses system data, analyzes the system data, creates updated system data, and replaces the system data in the second code section with the updated system data in response to the operation code; and,

wherein the system software is controlled to execute using the updated system data.

32. (Original) The system of claim 27 wherein the system data is stored in a third code section in the code storage section;

wherein the run-time engine accesses system data, analyzes the system data, creates updated system data, and replaces the system data in the third code section with the updated system data in response to the operation code; and,

wherein the system software is controlled to execute using the updated system data.

33. (Original) The system of claim 27 further comprising:

a volatile memory to accept read-write data;

wherein the run-time engine accesses the read-write data, analyzes the read-write data, creates updated read-write data, and replaces the read-write data in the volatile memory with the updated read-write data in response to the operation code; and,

wherein the system software is controlled to execute using the updated read-write data in volatile memory.

34. (Original) The system of claim 27 wherein the run-time engine monitors the execution of the system software, collects performance data, and stores the performance data in the file system section in response to the operation code; and,

wherein the system software is controlled to execute by collecting the performance data for evaluation of the system software.

35. (Original) The system of claim 34 wherein the run-time engine accesses the performance data from the file system section and transmits the performance data via an airlink interface in response to the operation code.

36. (Original) The system of claim 27 wherein the file system section receives a patch manager run time instruction including a new code section;

wherein the run-time engine adds the new code section to the code storage section in response to the operation code; and,

wherein the system software is controlled to execute using the new code section.

37. (Original) The system of claim 36 wherein the file system section receives a patch manager run time instruction including an updated code section;

wherein the run-time engine replaces a fourth code section in the code storage section with the updated code section in response to the operation code; and,

wherein the system software is controlled to execute using the updated code section.

38. (Currently Amended) In a wireless telephone ~~communications device~~, a dynamic instruction set execution system, the system comprising:

executable system software and system data differentiated into code sections with symbol libraries arranged within;

patch manager run time instructions (PMRTIs) organized as dynamic instruction sets with operation code and data items for operating on the system data and the system software, and for controlling the execution of the system software;

a file system section nonvolatile memory for receiving the patch manager run time instructions; and,

a run-time library arranged in a first code section for processing the dynamic instruction sets.